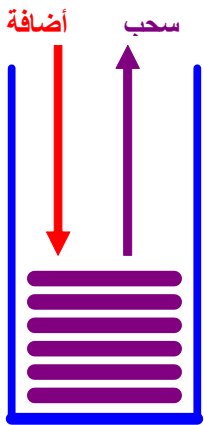


Stacks

A *stack* is a linear data structure which can be accessed only at one of its ends for storing and retrieving data. Such a stack resembles a stack of trays in a cafeteria: New trays are put on the top of the stack and taken off the top. The last tray put on the stack is the first tray removed from the stack. For this reason, a stack is called an *LIFO* structure: last in/first out.

A tray can be taken only if there are trays on the stack, and a tray can be added to the stack only if there is enough room, that is, if the stack is not too high. Therefore, a stack is defined in terms of operations which change its status and operations which check this status.



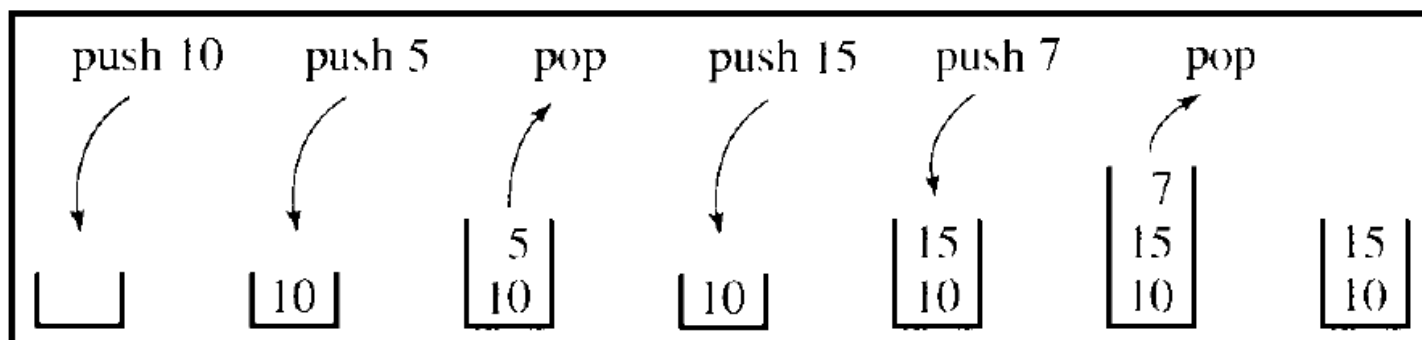
clear()—Clear the stack.

isEmpty()—Check to see if the stack is empty.

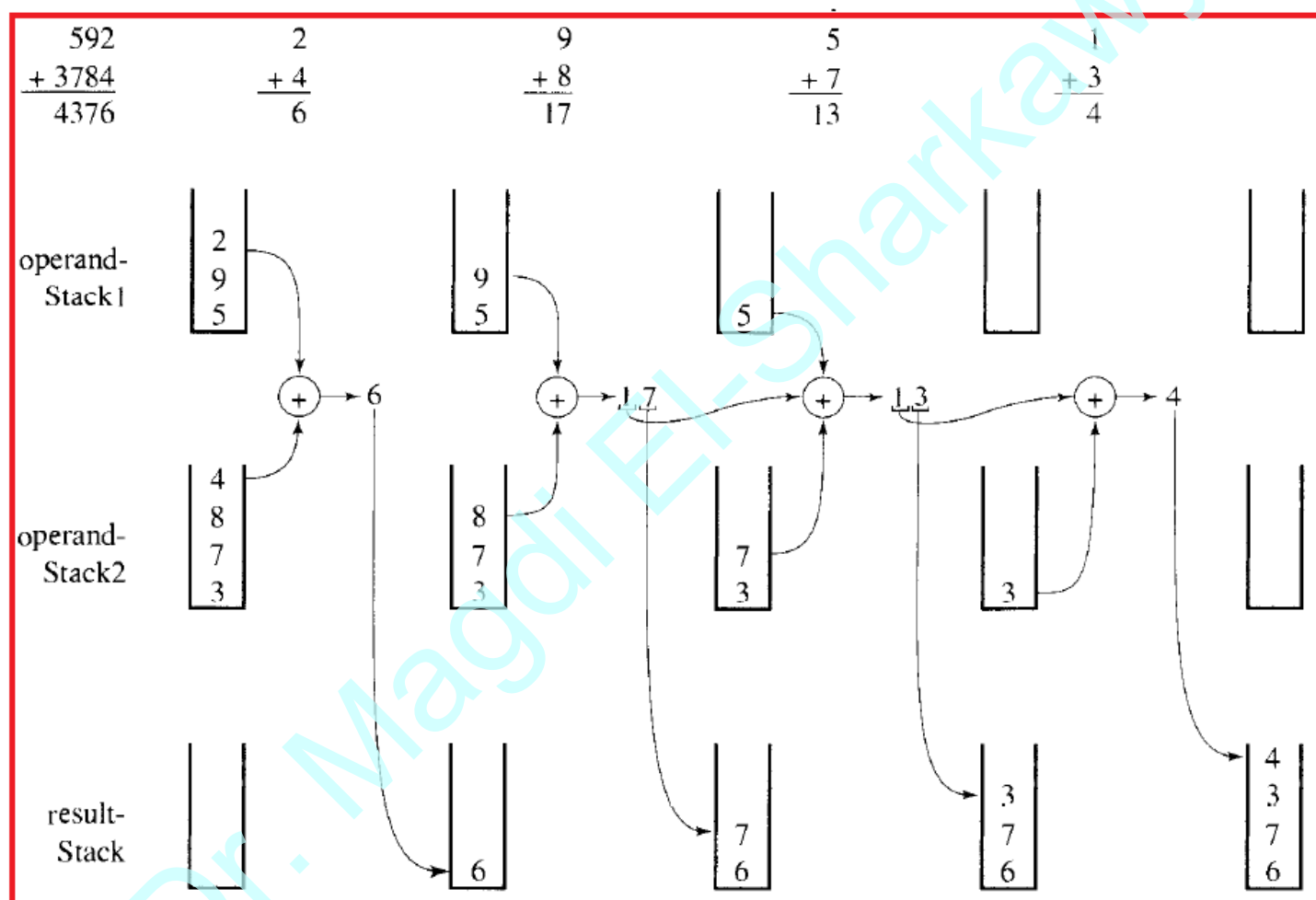
push(el)—Put the element *el* on the top of the stack.

pop()—Take the topmost element from the stack.

peek()—Return the topmost element in the stack without removing it.



أستخدام الـ Stack في اجراء العمليات الحسابية



Stack implementation (as a linked list):

```
// stack.cpp : by Dr. Magdi El-Sharkawy.
#include "stdafx.h"
#include <iostream>
using namespace std;
//-----class IntNode for creating new node -----//
class IntNode {
public :
    IntNode(int el, IntNode *ptr = 0) {info = el; next = ptr;}
    int info;
    IntNode *next;
};
//----- class stack for dealing with nodes -----//
class stack {
public:
    stack() {head = tail =0; }
    bool isempty();
    void push(int);
    int pop();
    int peek();
    void Displaystack();
    void clear();
private:
    IntNode *head, *tail;
};

bool stack::isempty()
{
    if (head==0 && tail ==0)
        return 1;
    else
        return 0;
}

void stack::push(int data)
{
    IntNode *newnode;
    newnode = new IntNode(data,0);
    newnode->next = head;
    head = newnode;
    if (tail==0)
        tail = head;
}
```

```

int stack::pop()
{
    int val=0;
    if(head!=0)
    {
        val = head->info;
        head = head->next;
    }
    if(head==0)
        tail=head;
    return val;
}

int stack::peek()
{
    if(!isempty())
        return head->info;
}

void stack::clear()
{
    head = tail =0;
}

void stack::Displaystack()
{
    IntNode *current;
    current = head;
    while(current != tail->next)
    {
        cout << current->info << "      " << current << "\n";
        current=current->next;
    }
    cout << "-----" << "\n";
}

void main()
{
    stack mag;
    if(mag.isempty()) cout << "Stack is empty \n";
    else cout << "Stack is not empty \n";
    mag.push(50);
    mag.push(90);
    cout << mag.peek() << "peeked \n";
    mag.push(60);
    mag.Displaystack();
}

```

```

    cout << mag.pop() << "    popped out \n";
    mag.Displaystack();
    mag.push(66);
    cout << mag.peek() << "peeked \n";
    mag.push(45);
    mag.push(38);
    mag.Displaystack();
    cout << mag.pop() << "    popped out \n";
    mag.Displaystack();
    mag.push(50);
    mag.Displaystack();
    mag.clear();
    if(mag.isempty()) cout << "Stack is empty \n";
        else cout << "Stack is not empty \n";
}

```

```

C:\Windows\system...
Stack is empty
90peeked
60    00291F10
90    000B19C8
50    000B1990
-----
60    popped out
90    000B19C8
50    000B1990
-----
66peeked
38    00291FB8
45    00291F80
66    00291F48
90    000B19C8
50    000B1990
-----
38    popped out
45    00291F80
66    00291F48
90    000B19C8
50    000B1990
-----
50    00291D00
45    00291F80
66    00291F48
90    000B19C8
50    000B1990
-----
Stack is empty
Press any key to continue

```